TYPESCRIPT-FUNKTIONEN DEKORIEREN

NICHT NUR ZU WEIHNACHTEN!

andrena
OBJECTS

ZIEL DIESER SESSION

- Bewusstsein für Decorators erhöhen
- Zeigen, wie man sie schreibt und verwendet
- Einige (hoffentlich) interessante Beispiele präsentieren
- Neugierde und Lust wecken, sie selbst auszuprobieren

WAS ERWARTET EUCH?

- Decorators im Allgemeinen
- Decorators in TypeScript
- Vorteile in Advent of Code
- Code, den ihr nie so verwenden solltet!
- Unsere eigenes kleines Dependency Injection Framework

WER BIN ICH?

Marco Sieben

- Fullstack-Entwickler mit einem besonderen Herzen fürs Frontend
- TypeScript-Enthusiast
- Liebt den jährlichen Advent of Code (=> im Dezember immer besonders müde)
- Hat dort Decorators erfolgreich eingesetzt

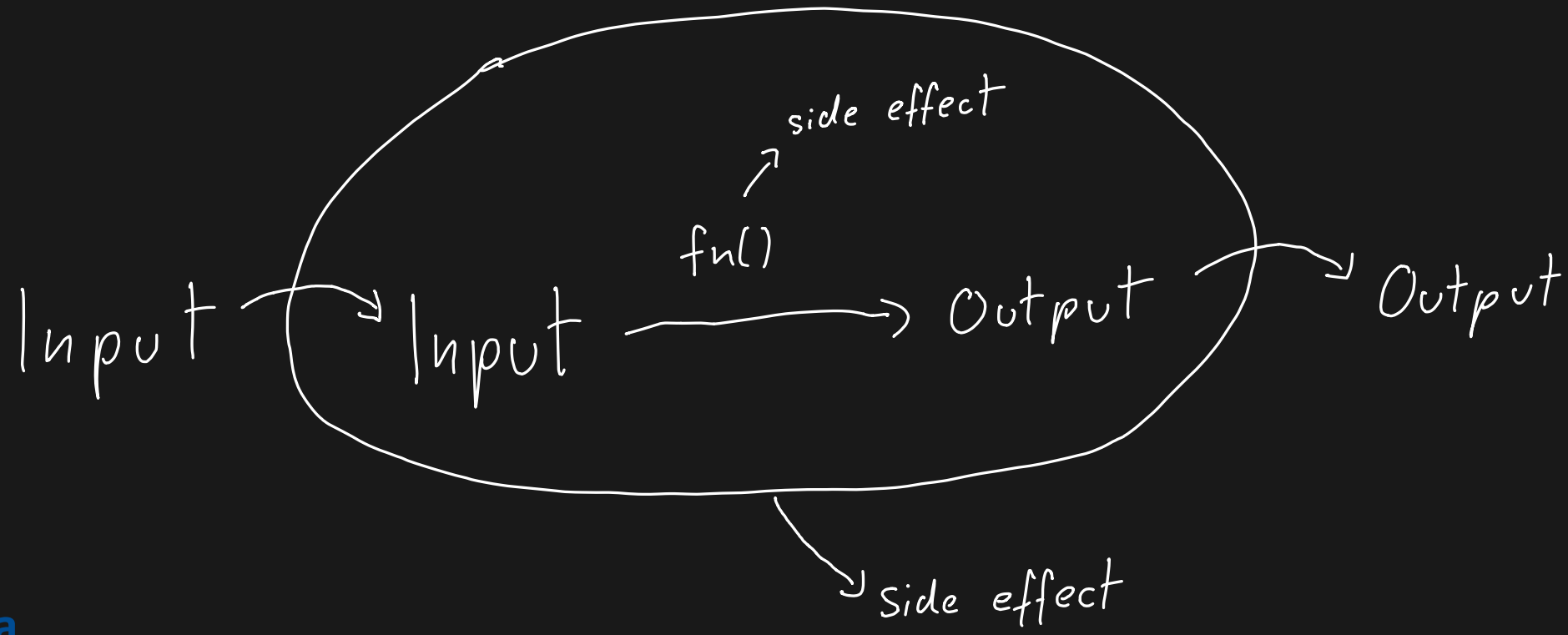
Danke an meinen Arbeitgeber andrena objects ❤️

andrena
OBJECTS

DECORATORS

DECORATORS

THEORIE



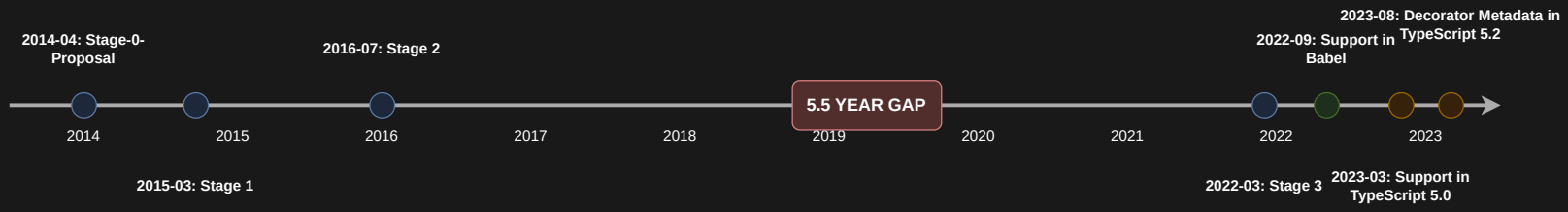
andrena
OBJECTS

DECORATORS

EINFACHES BEISPIEL

DECORATORS IN TYPESCRIPT

- Stufe-3-Vorschlag für ECMAScript => Irgendwann nativ in JavaScript
- Unterstützung in TypeScript seit TypeScript 5.0
- Ergänzungen/Anpassungen in 5.2 und 5.5



andrena OBJECTS

DECORATORS IN TYPESCRIPT

- Class method decorators
- Class decorators
- Class getter/setter decorators
- Class field decorators
- Class auto-accessor decorators

andrena
OBJECTS

CLASS METHOD DECORATORS

CLASS METHOD DECORATORS

BEISPIEL: LOGGED

CLASS METHOD DECORATORS

INTERFACE

```
type ClassMethodDecorator = (
  value: Function,
  context: {
    kind: 'method';
    name: string | symbol;
    metadata: object;
    static: boolean;
    private: boolean;
    access: { get: () => unknown };
    addInitializer(initializer: () => void): void;
  }
) => Function | void;
```

andrena
OBJECTS

ADVENT OF CODE

- Tägliche Herausforderungen (1. Dezember bis 25. Dezember)
- In eine durchgehende Geschichte eingebunden
- Keine Programmiersprache vorgegeben
- Globale und private Leaderboards
- Komplett kostenlos

Unbedingt angucken unter
https://adventofcode.com

andrena OBJECTS

CLASS METHOD DECORATORS

BEISPIEL 2: MEMOIZATION

CLASS DECORATORS

CLASS DECORATORS

INTERFACE

```
type ClassDecorator = (
  value: Function,
  context: {
    kind: 'class';
    name: string | undefined;
    metadata: object;
    addInitializer(initializer: () => void): void;
  }
) => Function | void;
```

CLASS DECORATORS

BEISPIEL: SINGLETON

CLASS DECORATORS

BEISPIEL 2: DEPENDENCY INJECTION

andrena
OBJECTS

WEITERE IDEEN

- Serialization
- Event Producer und Consumer
- Reactivity einführen
- ...

AUSBLICK

VERWANDTE THEMEN

- Andere Decorator-Typen (field, getter/setter, auto accessor)
- Decorator metadata
- auto accessors

LESEMATERIAL

- Release notes von TypeScript 5.0 (Decorators)
- Release notes von TypeScript 5.2 (Decorator metadata)
- Release notes von TypeScript 5.5 (Strikteres Parsing von Decorators)
- Der Decorator-Vorschlag für ECMAScript
- Der Decorator-Metadata-Vorschlag für ECMAScript
- Ausführlicher Blogpost von Axel Rauschmayer über den Decorators-Vorschlag

Repository mit Folien und Beispielen:

Marco Sieben

marco.sieben@andrena.de

https://www.linkedin.com/in/stnimmerlein

andrena OBJECTS

VIELEN DANK

UND VIEL SPASS BEIM DEKORIEREN

Repository mit Folien und Beispielen:

Marco Sieben

marco.sieben@andrena.de

https://www.linkedin.com/in/stnimmerlein

andrena OBJECTS